普通高等教育“十三五”规划教材

高等学校计算机规划教材

基于案例的软件构造教程

◆ 李劲华　周 强　陈 宇　编著

中国工信出版集团

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

普通高等教育“十三五”规划教材
高等学校计算机规划教材

基于案例的软件构造教程

李劲华　周　强　陈　宇　编著

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

本书以一个案例的演变模拟不断变化的用户需求，按照增量迭代的开发模式，将碎片化的功能开发、用户交互、数据处理等知识，以及软件设计、软件测试和敏捷开发的最佳实践，与软件开发的原理、技术和工具融合到设计、编码、调试及测试的构造过程。内容包括软件构造的一般原理（如模块化、增量迭代）、常用技巧（如表驱动编程、测试驱动开发）、软件设计（契约式设计、设计模式）、软件知识（如软件测试、软件复用）及软件构造的工作要素（如编码规范、构造工具）和活动（如设计、编码、测试、交付）。本书提供配套电子课件、案例源程序、例子代码、教学参考方案等。

本书面向计算机学科的本科学生，可作为“软件构造”、“Java 面向对象课程设计”等课程的教材，也可作为“实用软件工程”的参考书，同时也适合学习软件开发的其他专业及爱好者参考。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目 (CIP) 数据

基于案例的软件构造教程 / 李劲华，周强，陈宇编著. —北京：电子工业出版社，2016.6
高等学校计算机规划教材
ISBN 978-7-121-28272-0

I. ①基… II. ①李… ②周… ③陈… III. ①软件开发－教材 IV. ①TP311.52

中国版本图书馆 CIP 数据核字（2016）第 045030 号

策划编辑：王晓庆
责任编辑：王晓庆
印　　刷：涿州市京南印刷厂
装　　订：涿州市京南印刷厂
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×1092　1/16　印张：18.75　字数：517 千字
版　　次：2016 年 6 月第 1 版
印　　次：2016 年 6 月第 1 次印刷
印　　数：3000 册　　定价：45.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：(010) 88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：(010) 88254113，wangxq@phei.com.cn。

前　言

随着计算机与互联网在经济与日常生活中的渗透，各种形态的软件层出不穷，如传统的桌面软件、浏览器-服务器结构的 Web 应用、软件和服务 SaaS 及移动应用程序 APP 等，国家和社会对各类软件的需求不断增加。特别是移动 APP 的出现，再次凸显了小型软件开发的重要性。

传统计算机学科的课程体系涵盖了大量软件开发的知识，如高级语言程序设计、数据结构与算法、数据库、计算机网络、操作系统、编译原理、软件工程等，传授方式理论化、知识碎片化。设置和讲授这些课程的主要目的，甚至有些课程的唯一目的不是提升编程解决问题的能力，而是理解和研制计算机及系统软件。软件工程课程的核心作用是培养软件开发的需求分析、软件建模及团队协作、项目管理等实际工作需要的综合能力，但其基本前提是要求学生具备软件构造能力，即通过设计、编码、单元测试、集成测试和调试的组合，创建有用的软件。现代软件开发方法包括极限编程、测试驱动开发等敏捷方法，突出特点是每个程序员都具有高超的软件开发能力。他们不仅熟练掌握多种类型的编程语言、框架与中间件、设计模式、软件设计和测试技术及数据处理、用户交互等方面的知识，还要熟悉开发流程，具备把实际问题转换成软件的分析、设计和构造能力。

本书旨在以案例为引导，通过集成化解决软件知识的碎片化，提升个人的软件构造能力，加快从程序编写到软件开发的转变，在孤立的基础课程与软件工程课程之间搭建桥梁。目标是把学生培养成能独立地综合运用技术、开发可用产品的高级程序员，再通过后续课程（如软件工程、综合课程设计和实习实训）培养成软件工程师。

▶ 主要内容与结构

本书内容涵盖 IEEE 计算机学会最新颁布的“软件工程知识体系”中“软件构造”知识域的95%以上，以及软件设计、软件测试和敏捷开发的最佳实践，主要包括：

- 软件构造的一般原理：模块化，信息隐藏，逐步求精，面向对象原则，增量迭代，软件复用，软件质量。
- 软件构造的常用技巧：表驱动编程，防御式编程，按意图编程，事件驱动编程，代码重构，框架与程序包，测试驱动开发。
- 软件设计：软件建模及其语言 UML，E-R 图，控制流图，状态图，设计原则，设计模式，契约式设计，面向对象，用户交互，数据库的设计与实现。
- 软件知识：软件过程，敏捷开发，最佳实践，面向对象编程，数据结构与算法的实现，数据库编程，软件复用。
- 软件构造的工作要素：编码及其规范，构造工具如 IDE、Ant、JUnit。
- 软件构造的活动：设计，编码，调试，集成，测试（单元测试、回归测试、集成测试、静态测试）。

不同于传统软件教材按照开发活动或知识域的编写方式，本书以一个案例的演变，模拟不断变化的用户需求，以增量迭代的开发模式，编排这些教学内容。每章完成之后都有可用的、实现了用户要求功能或特性的程序。每章以案例故事引出构造问题，通过例子及设计和代码，讨论解

决问题的基本原理、方法、技术等最佳实践，给出一个可操作的构造方案，问题的可选设计、扩展则作为提升或留作练习。

全书共 9 章。第 1 章概述软件与软件开发的基本概念，说明软件构造的含义及其在整个软件开发过程中的地位和作用，引入本书案例。第 2～9 章以增量迭代方式，将功能开发、用户交互及数据处理等知识，与软件开发的原理、技术和工具融合到设计、编码、调试及测试的构造过程。第 2 章说明模块化概念及其软件的构造技术。第 3 章描述面向对象的设计原则及其软件的构造技术。第 4 章学习容器类数据结构与文件的使用和构造。第 5 章学习用户界面与软件集成的基础，说明非图形菜单式用户交互的设计与实现。第 6 章学习重构技术、自动化的软件打包与交付。第 7 章深入学习图形用户界面软件的构造。第 8 章学习数据库的设计并在应用程序中使用数据库。第 9 章学习软件复用，使用框架和程序包构造软件。

作者借鉴了国内外计算机科学与软件工程领域的研究与教学成果。首先是主编在德国斯图加特大学计算机系进修期间，其导师路德维希教授于 20 世纪末在德国大学开创性地设立了“软件工程”示范专业。其次是最近 10 多年我国软件工程领域教育先行者的大胆探索和实践。最后，在学科和专业技术方面对本书产生巨大影响的是“个体软件过程 PSP”、软件复用、敏捷开发方法及软件测试。在此对这些研究者和教育者表示崇高敬意和衷心感谢！

李劲华设计了本书的结构、内容及风格，编写了第 1～6 章、第 9 章内容及程序，审阅、校对了全书。周强编写了第 7 章内容及程序，陈宇编写了第 8 章内容及程序。

本书在编写过程中得到了学校、学院和电子工业出版社的大力支持，在此表示感谢！

本书谨代表作者们对软件构造的理解与探索。由于认识有限，加之时间仓促，难免存在疏漏和谬误，欢迎读者批评指正。可反馈给出版社（wangxq@phei.com.cn）或作者（E-mail：qduli@126.com，QQ：1487220149）。

▶ 使用建议

本书试图将软件构造的原理、原则、方法、技术、流程和技能整合，通过案例的展开由浅及深地学习。对知识的引入遵循足够、按需和渐进的原则。很多方面的知识（如调试、测试、设计、复用）分布在若干章节。读者最好把书中的知识通过案例串连起来，根据需要，可以跳过某个章节、提前看某个章节、需要时回过头来再看前面章节的内容或者查阅相关资料。建议读者一边思考案例问题，一边学习，一边动手实践。

就作者所知，目前国内外的课程体系中缺乏《软件构造》及相应的教材。本书可作为“软件构造”、“Java 面向对象课程设计”的教材，也可以选作部分专业面向实用的“软件工程”的参考书。用于“软件工程课程设计”、“软件工程综合实训”等课程时，需补充软件建模、项目管理、团队合作、应用领域等方面的知识。

受篇幅限制，书中对介绍的最佳实践仅给出了关键的实现代码，建议授课教师根据课时选择性采纳或改写。本书提供配套电子课件、案例源程序、例子代码、教学参考方案等，请登录华信教育资源网（http: //www.hxedu.com.cn）注册下载或联系本书编辑（wangxq@phei.com.cn）索取。

本书面向计算机学科的本科学生，也适合学习软件开发的其他学生及爱好者。建议在第 4～6 学期使用。要求读者具备程序设计和面向对象编程基础。有些内容可在需要时适当补充。对软件工程专业，后续课程可以是：软件工程、软件设计、软件测试、软件项目管理等。对计算机及其他应用程序设计技术的专业，建议选修“软件工程”或概论/导论，学习团队开发大型软件的技术和方法。

▶ 关于案例

案例的目的是模仿实际工作，传授软件构造的基本知识、主要活动、技术与工具的综合运用

此为试读，需要完整PDF请访问：www.ertongbook.com

能力。案例不宜太难，需求简单明了，使没有多少经验的学生容易理解，把重点放在软件的设计与构造，在一个学期就完成一个可运行的、有意义的程序。设计的案例产生3000行左右的代码，计划持续3个月、60人时的工作量。案例开发不指定某种具体编辑语言和软件过程。

案例是开发一个“50道100以内的加减法口算习题”的程序。具体要求模拟现实生活，在用户使用程序的过程中，不断提出新的要求和功能。

案例看似简单，却可以有多种理解方式及扩展。例如，100以内的运算数可包含100或0；可将运算数值扩大到500、1000；算式可以是至多包含两个运算符的加减法算术运算，还可以是至多包含4个运算符的四则算术运算。每章的提升与课后练习对案例提出了一些变化、延伸及构造，满足不同层次的教学要求。

本书程序主要采用Java语言。除个别情况外，书中不提供完整代码，意在培养学生独立解决问题的能力，构造不同的、可运行的软件。

▶ 教学方案

本书面向普通高校本科学生，提供3种教学参考方案，任课教师可更根据需要调整。例如，如果面向对象程序设计的基础较弱，可减少第2章学时；如果仅用GUI界面的用户交互，可减少第5章学时；如果学生程序设计能力较强，可减少第5章之前的实验，增加第9章学时。

（1）基本教学方案（方案1），72学时（18学时×5周），讲课38学时，实验34学时。

章	节	基本内容	讲课	实验内容	实验
1	掌握1.3节、1.5节和1.6节，理解1.1节、1.2节和1.4节，阅读案例1.7节	掌握程序与软件的异同、影响软件开发的因素、典型的软件开发过程、软件构造，理解软件的相关概念、软件生存周期和敏捷开发，阅读案例	3		
2	掌握2.2～2.4节、2.7.3节、2.7.4节，理解2.1节、2.5节、2.7.1节，阅读与实践2.6节，其余了解	掌握模块化设计、数据结构与算法的选择、测试设计、编码风格，理解模块化概念、测试概念、调试概念，阅读案例，其余了解	4	构造1、2	4
3	掌握3.2～3.5节，理解3.1节、3.7.2节、3.7.3节，阅读与实践3.6节，其余了解	掌握基本的面向对象设计技术、调试技术、测试框架，理解抽象、封装、ADT、面向对象的概念、设计模式、设计原则，阅读案例，其余了解	5	构造3	4
4	掌握4.2～4.4节、4.5.2节、4.6节、4.7.1～4.7.3节，理解4.1节、4.5.1节、4.7.4节，阅读与实践4.8节，其余了解	掌握文件/输入/输出流的编程、防御性编程、正则表达式、表驱动编程、白盒测试设计、JUnit其他测试，理解数据持久性、算式基及测试断言，阅读案例，其余了解	5	构造4、5、6	6
5	掌握5.1.4节、5.3节、5.4.3节、5.5节、5.7.1节、5.7.2节，理解5.1节其他、5.2节、5.4节，阅读与实践5.6节，其他了解	掌握菜单式用户交互的设计与编程、原型法、代码走查、静态分析工具、软件集成与测试、基于状态图的测试，理解用户交互的原则和开发过程、回归测试、静态测试，阅读案例，其余了解	5	构造7	4
6	掌握6.1.1节、6.2.2节、6.2.3节，理解6.1.2节、6.2.1节，阅读与实践6.3节、6.4.2节，其余了解	掌握基本的重构技术和过程、Java程序打包/交付，理解代码重构、软件交付，阅读案例和TDD，其余了解	4	构造8	4
7	理解7.1节、7.3节，掌握7.2节和7.4节，阅读与实践7.5节，了解7.6节	理解GUI的基本知识，掌握GUI的基本元素、设计规范及本书使用的Java GUI设计工具，掌握事件驱动编程方式，阅读案例，其余了解	4	构造9	4
8	理解8.1节、8.3节，掌握8.2节、8.4节、8.6.1～8.6.3节，阅读与实践8.5节，了解8.6.4节	理解数据库系统的结构，了解数据库的设计知识，掌握SQL的5种操作语句及数据库查询操作，掌握应用程序与数据库的连接和编程，阅读案例，其余了解	5	构造10	3
9	掌握9.3节，理解9.1节、9.2节、9.4节，阅读与实践9.5节，其余了解	掌握设计模式的编码实现、可视化显示、基于模板的文档产生，理解软件复用的概念、方式，阅读案例，其余了解	3	构造11、12、13	5
	合计		38		34

（2）面向对象技术的教学方案（方案 2），54 学时（18 学时×3 周），讲课 30 学时，实验 24 学时。

面向对象技术的教学方案

（3）Java 面向对象课程设计（方案 3），36 学时（18 学时×2 周），讲课 18 学时，实验 18 学时。

Java 面向对象课程设计

目　录

此为试读，需要完整PDF请访问：www.ertongbook.com

第 1 章　软件开发概述

从不同方面深入理解程序及其开发。首先，认识程序和软件的区别，了解不同的软件分类。其次，理解小程序的编写与大型软件的开发。理解软件的工程特征：从编写正确的程序，到开发高质量的软件；从注重编程语言、算法设计、数据结构等独立知识，到综合应用软件开发的原则、方法、技术和工具；从工艺到工程——分离软件开发的关注点，把它分成若干可以控制、可以管理的阶段或相关活动，以及这些活动的不同组合方式。最后，深入理解软件构造，从编辑-编译-运行的程序编写，到设计-实现-测试的软件构造。

1.1　程序与软件

1.1.1　从程序到软件

软件是由计算机程序和程序设计的概念发展演化过来的，是程序和程序设计发展到一定规模后并且在逐步商品化的过程中形成的。20 世纪 60 年代，随着计算机硬件的批量生产，工业界和学术界认识到了计算机程序的工程性和使用价值。一方面计算机程序必须随着硬件一起销售，向客户提供硬件不足于支持计算机的使用，即计算机程序具有复制价值。另一方面，计算机程序的开发过程不仅仅是上来就写程序，往往需要花费大量的时间厘清需求，花力气进行算法设计，在编程后还要对程序进行测试，以及向用户提供使用手册和文档，也就是说，计算机程序的生成是一种由多人合作、经历不同阶段的开发，且具有可复制和重复使用的器或件。

计算机程序（简称程序）是为了解决某个特定问题而用程序设计语言描述的适合计算机处理的语句序列。软件是能够完成预定功能和性能的可执行的程序和使程序正常执行所需要的数据，加上描述软件开发过程及其管理、程序的操作和使用的有关文档，即“软件=程序+数据+文档”。程序是软件的中枢和骨架，没有程序，软件就缺乏指挥，无法执行一系列指令完成预定的功能。数据是程序的血液，没有数据，软件就没有运行的驱动力，是一个没有筋骨的空壳子。文档赋予了软件可理解性、可用性和可操作性，否则，人们可能无法正常使用软件。

例如，一个教学管理软件，它通过一组独立而又交互的程序提供各种教与学的服务功能。学生用它登记选课，查阅课程安排、教师发布的信息和成绩等。教学管理人员使用该软件指派授课教师、安排教室及授课时间等。教师可以通过该软件查阅课程安排，发布课程信息、听课的学生信息，登录学生成绩等。该系统的数据具有多样性：有相对稳定的学生信息（如学号、姓名、专业、入学时间）、教师信息（如工号、姓名、院系、职称、简介）、课程信息（如编号、名称、内容简介、学时、学分）、教室信息（如编码、位置、配置）；有变动性强的数据，如授课信息（授课教师、时间和地点）；以及不断变化的数据，如授课教师发布的授课/参考材料、作业及批改、试题及分数、学生提交的作业、师生的答疑等。使用该软件不可或缺的还有各种文档，譬如如何安装、配置、启动程序的安装说明书，教务管理员、教师和学生如何使用软件完成各自任务的操作说明书，便于维护软件及其数据的软件结构、组成及数据字典、数据表等技术说明书。所有这些程序、数据和文档，共同构成了这个教学管理软件。

1.1.2 软件类型

软件按其功能划分为三种类型：系统软件、支撑软件和应用软件。系统软件负责管理计算机系统、网络及其各种独立的硬件，使它们可以协调工作，如计算机操作系统、设备驱动程序、通信处理、网络管理程序等。支撑软件是支持软件的开发、管理与维护的软件，如集成化开发环境（Integrated Development Environment，IDE）、编译程序、文件格式化程序、数据库管理系统（Database Management System，DBMS）、应用框架与程序库等。应用软件是为了某种特定的用途而开发的软件，种类形态最多，包括商业数据处理软件、工程与科学计算软件、计算机辅助设计/制造软件、系统仿真软件、智能产品嵌入软件、医疗/制药软件、管理信息系统、办公自动化软件、计算机辅助教学软件、游戏娱乐类软件、社交通信类软件等。

软件按其工作方式划分为：实时处理软件、分时软件、交互式软件和批处理软件。软件按服务对象的范围划分为：项目软件（定制开发）和产品软件（或通用软件）。软件的其他分类包括商业软件、开源软件、共享软件等。这些分类划分并不是互斥的。一个软件可能包含实时特性、分时特征，同时还具有交互性（如 Web 应用程序、即时通信软件）。一个产品可能要根据客户的特殊需求而作为项目进行客户化开发（如企业资源规划 ERP 软件），成熟之后打包成产品卖出。随着信息通信技术的发展，出现了软件开发与使用的新形态，如移动应用程序 APPs、平板应用程序及软件和服务（Software as a Service，SaaS）。

表 1.1 所示为一种按照软件规模划分的参考依据。可以看出，不同规模软件的代码行数可能相差极大。一个人完成一个大中型软件代码的编写是不现实的，比如受时间、成本、技术等因素的限制，人们不可能对所有软件都指望由一个天才的程序员完成开发。

表 1.1 软件按规模划分

类 别	参 加 人 数	研 制 期 限	产品规模（源代码行）
微型	1	1～4 周	0.5 千行
小型	1	1～6 月	1～4 千行
中型	2～5	1～2 年	5～50 千行
大型	5～20	2～3 年	50～500 千行
甚大型	100～1000	4～5 年	50 万～100 万行
极大型	2000～5000	5～10 年	200 万行及以上

随着软件变得越来越大、越来越复杂，软件开发的关注点也发生了变化，相对于小规模的程序设计（Programming in the Small），提出了大规模的程序设计（Programming in the Large），即软件开发。

1.1.3 程序设计与软件开发

计算机程序有两种形式：第一种是可执行程序，是能够在计算机中直接执行的指令集合；第二种是可读的源代码，经过翻译转换成可执行的程序。程序设计是解决特定问题而编写程序的过程，是软件生产活动中的重要组成部分。程序设计往往以某种程序设计语言为工具，编写源程序，然后由编译系统完成可执行代码的转换。程序设计的活动包括分析、设计、编码、测试、排错等不同阶段。

- 分析问题：研究所给定问题、条件，分析应达到的目标，找出解决问题的规律，定义问题，选择解题方法。

- 设计算法：设计出解题的方法和具体步骤（算法）。
- 编写程序：将算法翻译成计算机程序设计语言，然后对源程序进行编辑、编译和连接。
- 排错：运行可执行程序，得到运行结果。能得到运行结果并不意味着程序正确，还要对结果进行分析，看它是否合理。对程序进行调试，即通过上机发现和排除程序中的故障的过程。

程序设计是一种个人的科学或艺术。精美的算法和代码、程序的正确性是计算科学工作者所追求的主要目标。计算机程序的“科学和艺术”特征表现在程序的独创性、科学价值，以及不需要团队的重复劳动。图灵奖获得者沃斯（Niclaus Wirth）提出的“程序=算法+数据结构”是程序设计的精辟观点。数据结构指的是程序处理或应用的数据与数据之间的逻辑关系。算法指的是解决特定问题的步骤和方法。程序设计的核心就是选择和设计适合特定问题的数据结构与算法，用编程语言编制为程序。

另一位图灵奖获得者克努特（Donald Knuth）在其多卷书《计算机编程艺术》中展示了计算机编程的技巧和艺术性，并在其图灵奖演说中对计算机编程的艺术做了科学的论述和诠释。他和其他倡导者认为，编写优美的程序需要灵感和高超的技巧，就像诗人写诗、画家作画、建筑师构筑一样，充满了技巧、艺术和美感。计算机编程同样也是一门艺术，程序员就是创造这种艺术的艺术家。程序之美体现在精巧的算法机制、深邃的设计思想及优雅的代码布局。

在计算机技术发展的早期，软件开发的主要活动就是程序设计。随着程序向软件的演变，软件开发不再只是纯粹的程序的功能设计，还包括数据库设计、用户界面设计、软件接口设计、通信协议设计和复杂的系统配置。软件不再局限于科学研究，应用在工业、农业、银行、航空、军事、政府、教育、娱乐、社交等广泛领域。对于软件，人们不单纯地追求它正确地运行、完成预期的功能，还希望它可靠——在规定的条件和时间区间完成规定功能的能力，可用性——软件易学易用，能提高用户的体验和满足度，以及性能、安全等非功能性需求。

软件变得越来越大、越来越复杂。图 1.1 示意了 Linux 操作系统的内核在 10 年中从 240 万行扩大到接近 1600 万行。

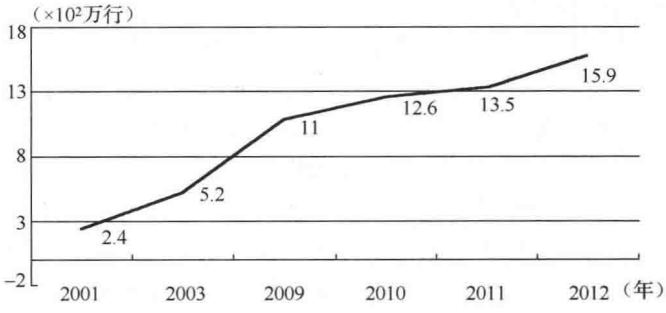

图 1.1　Linux 操作系统内核代码的变化

软件已经不可能仅仅依靠个人的才智、技巧手工地编写，而是需要团队的分工合作，采用工程化的方式，按照项目管理完成开发。程序设计活动走向软件工程。软件工程把经过时间考验而证明正确的管理技术和当前能够得到的最好的技术方法结合起来，以系统性的、规范化的、可定量的过程化方法去开发和维护软件。

软件工程是应用计算机科学、数学、逻辑学及管理科学等原理，开发软件的工程学科和活动。软件工程借鉴传统工程的原则和方法，以提高质量、降低成本及缩短开发时间。其中，计算机科学、数学用于软件模型与算法设计，工程科学用于制定规范、开发范型、评估成本及确定权衡，管理科学用于计划、资源、质量、成本等管理。软件工程不仅涉及程序设计语言、数据结构和算

法，还包括数据库、开发工具、人机交互、系统平台、设计模式等技术方面，以及项目管理、质量保障、标准规范、开发流程等管理因素。

1.2 软件生存周期

同任何事物一样，一个软件产品或软件系统也要经历孕育、诞生、成长、成熟、衰亡等阶段，一般称为软件生存周期（软件生命周期）。为了使规模大、结构复杂和管理复杂的软件开发或使用变得容易控制和管理，人们根据软件所处的状态和特征，把整个软件生存周期划分为若干阶段，每个阶段又包含不同的活动或任务。软件的生产者和使用者对软件生存周期的阶段和活动具有不同的理解和划分。

1.2.1 使用角度的软件生存周期

从用户的角度，软件生存周期分为下列三个阶段，每个阶段还可以进一步分为不同的途径或状态。

（1）提出需求。用户根据需求，提出要解决的问题和需要的软件。有时，用户甚至不清楚是否有软件或者能用软件在多大程度上解决这些问题；有时，用户可以描述要求的软件能力和使用预期；有时，用户能够清楚地说明软件的功能和非功能性需求（如性能、安全性、可用性）、运行环境、基础设施及约束条件等。

（2）获取软件。这个过程主要是对获取软件的最佳途径做出决策并选择最佳的供应商。软件的获取有三种主要途径。①购买软件，像购买其他商品一样，用户可以购买符合用户需求的商品软件（Commercial-off-the-shelf，COTS）。但是，对于软件，通常购买的是软件使用授权，而不是软件本身，即用户没有软件的所有权。②定制或开发软件，又可以分为三种形式：外包开发——由独立软件开发商按照用户的需求开发软件，用户可以拥有软件所有权；独立开发——用户自己或用户组织内部的专业人员按照需求开发软件；联合开发——用户与独立软件开发商采取联合的方式共同开发用户需要的软件，双方可能分享软件的所有权。③租赁软件或租赁服务，用户不占有软件本身，甚至无须在用户现场安装软件系统，而是通过客户端（如手机客户端软件、Web 浏览器）使用软件提供的操作或服务，例如，云计算的一种形式“软件及服务”（SaaS）。

（3）使用软件。一旦获得软件之后，用户将操作软件使之为其服务。例如，使用社交软件联系沟通、使用银行软件处理银行业务。当软件有了新的版本时，可自动或手动更新软件。当然，如果是租赁的软件或者采用了云服务，则由软件提供商完成软件的更新维护，用户无须做任何事情。对于大型、复杂的软件系统（如企业资源规划 ERP、电子政务系统），通常要通过软件实施才能使用软件，即通过开发商或其服务商在用户现场搭建运行环境，安装并配置软件、培训用户、录入基础数据，才能使用软件。最后，如果不需要了，就废弃正在使用的软件，或者更换其他软件。

1.2.2 开发角度的软件生存周期

从软件开发者的角度，一般把软件生存周期分为定义软件、开发软件和维护软件三个阶段。表 1.2 所示为这三个阶段的主要任务。

软件生存周期的一些活动构成了软件开发生命周期，即从决定开发软件产品到交付软件产品。这个过程主要包括需求阶段、设计阶段、实现阶段、测试阶段，有时还包括安装阶段。根据采用的软件开发方式，这些阶段的划分会重复执行。

表 1.2　软件生存周期

阶　段	活　动	描　述
定义软件	理解问题	用明确的语言描述软件要解决的问题、目标和范围
	可行性研究	从经济、技术、法律等方面分析软件开发的可行性
	需求分析	描述对软件系统的所有需求，即明确要软件做什么
开发软件	软件设计	建立目标软件的解决方案，包括软件的结构和组成
	软件实现	用程序语言实现设计方案，包括与其他系统的集成
	软件测试	通过各种测试和评审技术，确认软件满足指定要求
维护软件	软件交付	发布开发的软件，或者安装、部署到用户现场以便使用
	软件维护	对软件进行修改或对需求的变化做出响应
	软件退役	终止对软件的技术支持和维护，软件停止使用

1. 需求定义

一旦决定了开发软件，就要研究用户需求、确定软件的定义，通常是功能需求和特性需求。功能需求定义软件在抽象级别应该提供的基本功能。详细的功能需求说明应该在子系统级别完成。例如，对于教学管理软件，需求分析活动包括识别存储和管理课程信息的数据库，识别教师、学生和教务管理者对系统的不同操作。非功能性需求指的是软件应该具备的特性，如可用性、性能、安全性、可靠性等。它们适合并影响所有的子系统。

此外，需求定义阶段的一个重要部分是建立一组系统应当满足的总体目标。这些目标可能会拓展、也可能会限制设计的抉择。例如，对上述系统要求，未经授权，任何人不能修改已经安排好的课程信息。

2. 软件设计

软件设计的一般涵义是指为了达到某一特定目的，通过制定计划、进行思考与概念的组织，建立一个现实可行的方案，并用设计语言明确地表示出整个过程。

软件设计是给出如何实现需求的决策和方案，是将系统功能分配到系统不同组成元素的过程，包括一组活动。①划分需求：对需求进行分析并合并成相关的组。通常有几种可能的划分方式，在这个阶段要做出选择。②确定子系统：识别出独立或集体满足需求的子系统。成组的需求通常与子系统相关，这个活动可以和划分需求一起完成。③给子系统分配需求：原则上，只要子系统的识别是需求划分驱动的，这个任务就直截了当。但是，在实践中，需求划分和子系统识别从来都不是完全匹配的，使用的外部系统或库函数可能会要求更改需求。④定义子系统的功能：确定每个子系统或模块的特殊功能。在这个阶段也应该确定各个子系统之间的关系。⑤定义子系统的接口：定义每个子系统提供的或需要的接口。一旦确定了这些接口，就可以同步开发各个子系统了。

设计过程的不同活动之间存在大量反馈和循环。随着问题和疑问的出现，常常要对早期的工作返工。

软件可以在不同层次和方面进行设计。经典的软件工程把软件设计分为概要设计和详细设计。按照软件的组成，可以进行系统架构设计、子系统及构件设计、接口设计及算法设计和数据结构的程序设计。另外，软件还可以进行功能设计、用户界面设计和数据库设计。

3. 软件实现

软件实现是完成可以运行程序及数据的软件开发过程。开发者用合适的编程语言（包括数据库语言、Web 语言、脚本语言等）把软件的设计转换成程序，包括编码、调试和测试活动。软件

系统通常有不同的组成，在设计时可能抽象成逻辑上的子系统、模块或实体、关系和属性等。在实现阶段，要把这些逻辑元素表示成函数、类及其以及记录、表和字段等。对设计阶段的用户交互则可以实现成用户的输入（键盘、鼠标等）、屏幕等输出及菜单、按钮、窗口等对象。

软件实现与通常的程序设计的主要区别是软件的集成与测试。软件集成指的是通过函数调用、消息传递、事件响应、状态改变、服务合成等机制把编程实现的各个软件单元组装在一起，形成一个更大的软件单元或可以运行的软件系统。有时，需要编写一些脚本来连接、启动数据库、运行环境等，集成其他外部资源。软件测试也属于实现阶段的活动，可以分为对程序基本组成单元的测试（单元测试）、对软件组装结果的测试（集成测试）、对整个软件系统的测试（系统测试），以及把软件交付给用户时的测试（验收测试）。

4．软件维护

软件维护是指对已完成开发并发布、交付使用的软件产品进行完善、纠正错误、改进性能和其他属性，或使软件适应改变了的环境。软件维护分为 4 种类型。

（1）改正性维护：修正出现的软件缺陷。软件交付使用后，会有一部分隐藏的错误被带到运行阶段，这些隐藏的错误在某些特定的使用环境下就会暴露，必须消除。

（2）适应性维护：为适应环境的变化而进行的软件修改活动。

（3）完善性维护：为满足不断变化的需求而改善和加强产品的功能与性能。

（4）预防性维护：为改善软件系统的可维护性和可靠性，也为以后的改进奠定基础。

在软件生存周期中，软件维护会持续很长一段时间，甚至超过全周期的一半。良好的软件开发过程和技术（如软件复用、模块化、面向对象、建模）是保证软件质量、减少维护工作的基础。为了使软件易于维护，开发时就必须考虑使软件具有可维护性。软件可维护性可通过三个质量特性来衡量：可理解性，指维护人员理解软件的结构、接口、功能和内部过程的难易程度；可测试性，指测试和诊断软件错误的难易程度；可修改性，指修改软件的难易程度。

软件再工程和代码重构是在软件维护中经常使用的技术。

1.3 软件开发过程

早期软件开发的活动主要就是编码，没有明确定义的工作流程，通常是边写代码边修改，以便快速地交付可运行程序。这种开发方式没有细致地分析需求、详细地设计软件、系统地测试软件或软件单元，对整个软件的开发缺乏计划和控制，难以保障开发软件的质量。人们借鉴了桥梁、电气、建筑等工程领域，把软件开发工作分成若干都有特定结果的步骤或活动，按照一定的方式开展软件活动，从而形成了软件过程。

根据每个活动的内容、活动的执行顺序、是否多次执行、活动参与者（特别地，是否包含客户/用户）、使用何种工具、采取哪些管理、活动产出等，出现了若干不同的软件开发过程。简单而言，软件开发过程是用来生产软件产品的流程及一系列工具、方法和实践的集合。软件过程模型是从一个特殊的视角对软件开发过程的简化描述。模型的本质就是简化。所以，软件开发过程模型是对实际过程的抽象。

图 1.2 所示为一个典型的软件开发过程。从软件的生产过程看，编写程序，甚至包括系统分析与设计的软件开发也只是其中的一部分工作。除了最上面的软件生存周期的开发活动，还有技术支持、文档编写、测试开发及过程管理等非生产性活动，该图同时示意了在软件不同的生存周期，各种活动对资源（人、成本等）的需要是不均衡的。软件实现阶段占用了绝大部分资源。

此为试读，需要完整PDF请访问：www.ertongbook.com